课程回顾

- NP完全性理论
  - 归约
  - NPC/NP-Hard问题形式化定义
  - NP完全性理论
  - 典型的NPC问题（合取范式可满足性问题、团问题、顶点覆盖问题、子集和问题、哈密顿回路问题、旅行商问题）
- 递归式（举例：阶乘、斐波那契数列、Ackermann函数、汉诺塔问题、全排列问题）

递归式与分治法

- 递归式与分治方法紧密相关，使用递归式可以很自然地刻画分治算法的运行时间
- 分治算法设计：将一个问题分解为与原问题相似但规模更小的若干子问题，递归地解这些子问题，然后将这些子问题的解结合起来构成原问题的解。这种方法在每层递归上均包括三个步骤：
  - 分解（Divide）：将问题划分为若干个子问题
  - 解决（Conquer）：递归地解这些子问题；若子问题规模足够小，则直接解决
  - 合并（Combine）：将子问题的解组合成原问题的解

递归式与分治法 (续)

■求解递归式方法（得出算法Θ或O渐近界）：

- 代入法：猜测一个界，然后用数学归纳法证明这个界是正确的
- 递归树法：将递归式转换为一棵树，结点表示不同层次的递归调用产生的代价，然后采用边界和技术来求解递归式
- 主方法：可求解形如$T(n)=aT(n/b)+f(n)$递归式的界

归并排序

- 分解：分解待排序的n个元素的序列成各具$n/2$个元素的两个子序列
- 解决：使用归并排序递归地排序两个子序列
- 合并：合并两个已排序的子序列以产生已排序的答案
  - 重复以下步骤直至两个子序列没有未处理的元素：
    - 比较两个子序列中第一个未处理的元素
    - 将较小的元素复制到输出序列A中，将相应子序列下标后移一位表示该元素已被处理过
  - 当一个子序列中的所有元素都被处理过后，复制另一个子序列中所有元素至A中

归并排序

■分解：分解待排序的n个元素的序列成各具$n/2$个元素的两个子序列

人们从大量实践中发现，在用分治法设计算法时，最好使子问题的规模大致相同。即将一个问题分成大小相等的k个子问题的处理方法是行之有效的。这种使子问题规模大致相等的做法是出自一种平衡(balancing)子问题的思想，它几乎总是比子问题规模不等的做法要好。

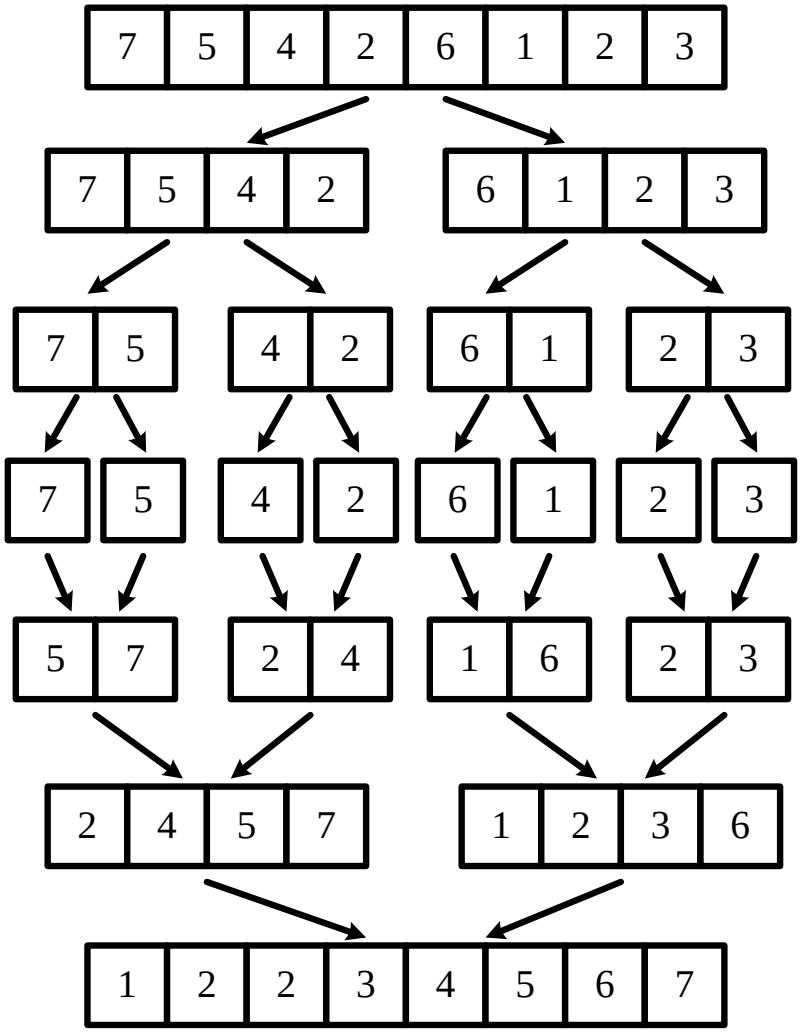
归并排序 (续)

```
MERGE_SORT(A, p, r)
1  if p < r
2      q ← ⌊(p + r)/2⌋
3      MERGE_SORT(A, p, q)
4      MERGE_SORT(A, q+1, r)
5      MERGE(A, p, q, r)
```

```
MERGE(A, p, q, r)
1  n1 ← q – p + 1
2  n2 ← r – q
3  let L[1..n1+1] and R[1..n2+1] be new arrays
4  for i ← 1 to n1 do L[i] ← A[p+i–1]
5  for j ← 1 to n2 do R[j] ← A[q+j]
6  L[n1+1] ← ∞;  R[n2+1] ← ∞
7  i ← 1;  j ← 1
8  for k ← p to r do
9      if L[i] ≤ R[j]
10         A[k] ← L[i];  i ← i + 1
11     else A[k] ← R[j];  j ← j + 1
```

所有情形时间复杂度均为$\Theta(n\lg n)$　　空间复杂度为$O(n)$

归并排序 (续)


7 5 4 2 6 1 2 3
7 5 4 2
6 1 2 3
7 5
4 2
6 1
2 3
7 5
4 2
6 1
2 3
5 7
2 4
1 6
2 3
2 4 5 7
1 2 3 6
1 2 2 3 4 5 6 7

分治算法时间性能分析

- （教材p20）设$T(n)$是输入规模为n的执行时间，若规模足够小，如$n \le c$ (常数)，则直接求解的时间为$\Theta(1)$
  - 设完成划分的时间为$D(n)$
  - 设分解时，划分为a个子问题，每个子问题规模为原问题的$1/b$，则解各子问题的时间为$aT(n/b)$
  - 设合并时间$C(n)$

$$T(n) = \begin{cases} \Theta(1), & n \le c, \\ aT(n/b) + D(n) + C(n), & \text{otherwise.} \end{cases}$$

边界

$b>1$，否则无限递归

例如归并排序中：$a=2$，$b=2$，$D(n)=O(1)$，$C(n)=\Theta(n)$

分治算法时间性能分析 (续)

```
MERGE_SORT(A, p, r)
1  if p < r
2     q ←⌊(p + r)/2⌋
3     MERGE_SORT(A, p, q)
4     MERGE_SORT(A, q+1, r)
5     MERGE(A, p, q, r)
```

■归并排序的最坏情况运行时间

$$T(n) = \begin{cases} \Theta(1), & n = 1, \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n), & n > 1. \end{cases}$$

$$T(n) = \begin{cases} \Theta(1), & n = 1, \\ 2T(n/2) + \Theta(n), & n > 1. \end{cases}$$

$$T(n) = 2T(n/2) + \Theta(n)$$

■（教材p38）一般地，解递归式时可忽略细节

- 假定函数参数为整数，如n为奇数时两个子问题规模为$\lfloor n/2 \rfloor$和$\lceil n/2 \rceil$
- 边界条件可忽略，当n较小时$T(n)=\Theta(1)$

■因为这些细节一般只影响常数因子的大小，不改变量级。求解时，先忽略细节，然后再决定其是否重要

以下我们首先讨论细节问题！

递归式求解——代入法

- （教材p47-49）代入法求解递归式分为两步：
  - 猜测解的形式
  - 用数学归纳法求出解中的常数，并证明解是正确的
- 关键：用猜测的解代入到递归式中
- 例：确定$T(n) = 2T(\lfloor n/2 \rfloor) + n$的上界
  - 猜测解$T(n)=O(n\lg n)$，需要证明恰当选择常数$c>0$，有$T(n)\leq cn\lg n$

递归式求解——代入法 (续)

■例：确定$T(n) = 2T(\lfloor n/2 \rfloor) + n$的上界

- 猜测解$T(n)=O(n\lg n)$，需要证明恰当选择常数$c>0$，有$T(n)\le cn\lg n$
- 先看此猜测是否正确：假定该上界对所有正数$m<n$都成立，特别是对于$m = \lfloor n/2 \rfloor$有

$$T(\lfloor n/2 \rfloor) \le c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)$$

代入递归式得到：

$$\begin{aligned} T(n) &\le 2c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor) + n \le cn\lg(n/2) + n \\ &= cn\lg n - cn\lg 2 + n \\ &= cn\lg n + (1-c)n \le cn\lg n \end{aligned}$$

只要$c\ge 1$即可

递归式求解——代入法 (续)

■例：确定$T(n) = 2T(\lfloor n/2 \rfloor) + n$的上界

- 猜测解$T(n)=O(n\lg n)$，需要证明恰当选择常数$c>0$，有$T(n)\le cn\lg n$
- 数学归纳法要求证明边界条件成立：假设$T(1)=1$是递归式唯一边界条件，但$T(1)\le c\cdot 1\cdot \lg 1=0$并不成立
- 渐近符号仅要求对$n\ge n_0$证明$T(n)\le cn\lg n$
  - 扩展边界条件
  - 令$n_0=2$，之前的证明成立需要对$n_0 \le m = \lfloor n/2 \rfloor < n$都成立，因此数学归纳法中需满足$n\ge 4$，此时只要找到足够大的$c$使得$T(2)$和$T(3)$满足$T(n)\le cn\lg n$即可
  - $T(2)=4$，$T(3)=5$，即任何$c\ge 2$即可满足

递归式求解——代入法 (续)

- 注意事项1：做出好的猜测（没有一般方法，只能凭经验）
  - 与见过的解类似，则相应猜测

    例：$T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$

    当n足够大时，$\lfloor n/2 \rfloor$与$\lfloor n/2 \rfloor + 17$相差无几，可猜测$T(n)=O(n\lg n)$
  - 先证较宽松的上下界，减小猜测范围

    例：$T(n) = 2T(\lfloor n/2 \rfloor) + n$

    显然$T(n)=\Omega(n)$　//式中有n相关的项

    　　$T(n)=O(n^2)$　//最多分解$O(n)$次，每次执行时间为n

    逐渐降低上界提升下界，直至收敛到渐近紧确界$T(n)=\Theta(n\lg n)$

递归式求解——代入法 (续)

■注意事项2：细节修正

- ➢有时猜测解是正确的，但数学归纳法却不能直接证明其细节
- ➢可从猜测解中减去一个低阶项使数学归纳法满足

例：$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$

猜测$T(n)=O(n)$，即需证明某个常数$c>0$，$T(n)\leq cn$成立

将猜测代入递归式得到$T(n) \leq c(\lfloor n/2 \rfloor) + c(\lceil n/2 \rceil) + 1 = cn + 1$

并不能得出$T(n)\leq cn$的结论

减去一个低阶项，新猜测为$T(n)\leq cn-d$ ($d\geq 0$为常数)

$T(n) \leq (c\lfloor n/2 \rfloor - d) + (c\lceil n/2 \rceil - d) + 1 = cn - 2d + 1 \leq cn - d$

只要$d\geq 1$即可。c通过边界条件选择